Backing up MySQL can be a chore, especially doing it manually. It's a good thing, Ryan Haynes posted a short script that should automate backing up in the MySQLDump reference manual page , it was further enhanced by Bill Hernandez in the same post. I have added my own tweaks to the script, and am using it to backup my MySQL databases. Read on to get the script.

You can just copy and paste the script to a file and chmod +x bashfile.sh to make it executable.

To add your cron (automatic execution), just use crontab -e and add the line 0 8 * * * /directory/bashfile.sh (this will automatically run the script every 8 a.m.)

Here is my version of the script. For the original scripts go here .

Get a copy of the script at the download section or just copy and paste.

------ start of script (copy below) ------

```
#!/bin/bash

  # SEE : http://dev.mysql.com/doc/refman/5.0/en/mysqldump.html  # SEE :
http://safari.oreilly.com/0596526784/date_and_time_string_formatting_with_strftime

#

# Additional tweaks/refinement by Francis Manas  on May 26, 2010

# ( 1 ) Separated dumping and compression, to allow finishing of dumping of all databases first
before compressing

# ( 2 ) Rearranged log file to include processing for easier determination where it stopped.

# ( 3 ) Added host_computer to MySQL credentials

# ( 4 ) Comment out asking of sudo password to be able to include in crontab (but you should
have permission to access backup directory)

# ( 5 ) Added options for compression

# ( 6 ) Added check to limit size of log file

# ( 7 ) Added RSYNC option to secondary drive

# Improved by Bill Hernandez (Plano, Texas) on Tuesday, August 21, 2007 (12:55 AM)

# ( 1 ) Backs up all info to time stamped individual directories, which makes it easier to track

# ( 2 ) Now maintains a single log that contains additional information

# ( 3 ) Includes a file comment header inside each compressed file

# ( 4 ) Used more variables instead of hard-code to make routine easier to use for something
else

# ( 5 ) Where I have mysql, you may have to replace it with mysql

#

# Posted by Ryan Haynes on July 11 2007 6:29pm

#

# DO NOT DELETE AUTOMATICALLY FOR NOW, MAYBE LATER
```

```
DELETE_EXPIRED_AUTOMATICALLY="TRUE"  # VALID VALUES "TRUE"/"FALSE"

# RSYNC TO SECONDARY FOLDER (ie. 2nd Drive)

# Note: This is currently set to a locally mounted drive/directory

#       For remote access, rsync command must be customized to needs.

ACTIVATE_RSYNC="TRUE"  # VALID VALUES "TRUE"/"FALSE"

# STORE NUMBER OF BACKUPS TO RECORD IN LOG BEFORE RESET

COUNT_BACKUP_LOG=5

# ENTER BACKUP DESTINATION

BASE_DIR=~/Backup/MySQL

RSYNC_DIR=~/RSync/Folder

# ENTER MYSQL CREDENTIALS

mysql_username="mysqluserid"

mysql_password="mysqlpassword"

host_computer="mysql.host.com"

# ACTIVATE COMPRESSION

ACTIVATE_COMPRESSION="TRUE"  # VALID VALUES "TRUE"/"FALSE"

# COMPRESSION METHOD

# compression_method="bzip2"

compression_method="gzip"

# DELETE EXPIRED BACKUPS THAT ARE MORE THAN

expire_minutes=$(( 1 * 2 )) # 2 minutes old for testing only

# expire_minutes=$(( 1 * 30 )) # 30 minutes old

# expire_minutes=$(( 60 * 24 )) # 1 day old
```

```
# expire_minutes=$(( 60 * 24 * 7 )) # 7 days old

# expire_minutes=$(( 60 * 24 * 30 )) # 30 days old

# expire_minutes=$(( 60 * 24 * 7 )) # 7 days old

if [ $expire_minutes -gt 1440 ]; then

expire_days=$(( $expire_minutes /1440 ))

else

expire_days=0

fi

#function pause(){

#read -p "$*"

#}

# pause "HIT RETURN, and then enter your sudo password..."

# echo "Please enter your sudo password..."

# sudo echo

current_dir=`pwd`

echo -n "Current working directory is : "

echo $current_dir

echo "------------------------------------------------------------------------"

TIME_1=`date +%s`

TS=$(date +%Y.%m.%d-%I.%M.%p)

BACKUP_DIR=${BASE_DIR}/$TS

BACKUP_LOG_NAME=mysql_dump_runtime.log

BACKUP_LOG=${BASE_DIR}/${BACKUP_LOG_NAME}
```

```
mkdir -p $BACKUP_DIR

#sudo chown mysql:admin $BACKUP_DIR

chmod 775 $BASE_DIR

chmod -R 777 $BACKUP_DIR

cd $BASE_DIR

# check if backup log is too long

if [  -f $BACKUP_LOG ]; then

countBackups=`grep -c "Start backup process" "${BACKUP_LOG}"`

echo $countBackups

if [ $countBackups -ge $COUNT_BACKUP_LOG ]; then

rm -R $BACKUP_LOG

fi

fi

if [ ! -f $BACKUP_LOG ]; then

echo "--------------------------------------------------------------------------------" > ${BACKUP_LOG_NAME}

echo "THIS IS A LOG OF THE MYSQL DUMPS..." >> ${BACKUP_LOG_NAME}

echo "DATE STARTED : [${TS}]" >> ${BACKUP_LOG_NAME}

echo "--------------------------------------------------------------------------------" >> ${BACKUP_LOG_NAME}

echo "[BACKUP DIRECTORY ]   [PROCESS DETAILS]" >> ${BACKUP_LOG_NAME}

echo "--------------------------------------------------------------------------------" >> ${BACKUP_LOG_NAME}

countBackups=0

fi

countBackups=$(( countBackups + 1))
```

```
echo "[${TS}] Start backup process #$countBackups." >> ${BACKUP_LOG_NAME}

echo "[${TS}] Saving mysql dumps to '$BACKUP_DIR'." >> ${BACKUP_LOG_NAME}

cd $BACKUP_DIR

echo -n "Changed working directory to : "

pwd

echo "Saving the following backups to '$BACKUP_DIR'."

echo "------------------------------------------------------------------------"

# get databases from list

DBS="$(mysql --host=${host_computer} --user=${mysql_username}
--password=${mysql_password} -Bse 'show databases')"

#  Process dump of databases

for db in ${DBS[@]}

do

# create sql filename

normal_output_filename=${db}.sql

if [ $compression_method == "gzip" ]; then

compressed_output_filename=${normal_output_filename}.gz

else

compressed_output_filename=${normal_output_filename}.bz2

fi

cd $BASE_DIR

echo "[${TS}] Now dumping database '$db' to '$normal_output_filename'." >>
${BACKUP_LOG_NAME}

cd $BACKUP_DIR
```

```
echo "-- SQL File: $normal_output_filename - $TS" > $normal_output_filename

echo "-- Compressed File: $compressed_output_filename - $TS" >> $normal_output_filename

echo "-- Logname  : `logname`" >> $normal_output_filename

echo "Now dumping '$db' to '$normal_output_filename'."

# Modify mysqldump to include your specific parameters

mysqldump --host=${host_computer} --user=${mysql_username}
--password=${mysql_password} $db --single-transaction -R >> $normal_output_filename

cd $BASE_DIR

echo "[${TS}]     Dumped database '$db' to '$normal_output_filename'." >>
${BACKUP_LOG_NAME}

done

echo "------------------------------------------------------------------------"

if [ $ACTIVATE_COMPRESSION == "TRUE" ]; then


#  Process compression separate from the dump.

for db in ${DBS[@]}

do


normal_output_filename=${db}.sql

if [ $compression_method == "gzip" ]; then

compressed_output_filename=${normal_output_filename}.gz

else

compressed_output_filename=${normal_output_filename}.bz2

fi


cd $BASE_DIR
```

```
echo "[${TS}] Now compressing '$normal_output_filename' to '$compressed_output_filename'."
>> ${BACKUP_LOG_NAME}

cd $BACKUP_DIR

echo "Now compressing '$normal_output_filename' to '$compressed_output_filename'."

# add custom parameters to your compression method

if [ $compression_method == "gzip" ]; then

gzip -7 -c $normal_output_filename > $compressed_output_filename

else

bzip2 -c $normal_output_filename > $compressed_output_filename

fi


rm $normal_output_filename

cd $BASE_DIR

echo "[${TS}]     Compressed '$normal_output_filename' to '$compressed_output_filename'." >>
${BACKUP_LOG_NAME}

done

echo "-------------------------------------------------------------------------"

fi

cd $BASE_DIR

echo -n "Changed working directory to : "

pwd

echo "Making log entries..."

# delete old databases. I have it setup on a daily cron so anything older than 60 minutes is fine

if [ $DELETE_EXPIRED_AUTOMATICALLY == "TRUE" ]; then
```

```
counter=0

for del in $(find $BASE_DIR -name '*-[0-9][0-9].[0-9][0-9].[AP]M' -mmin +${expire_minutes})

do

counter=$(( counter + 1 ))

echo "[${TS}] [Expired Backup - Deleted] $del" >> ${BACKUP_LOG_NAME}

done

echo "------------------------------------------------------------------------"

if [ $counter -lt 1 ]; then

if [ $expire_days -gt 0 ]; then

echo There were no backup directories that were more than ${expire_days} days old:

echo "[${TS}] There were no backup directories that were more than ${expire_days} days old."
>> ${BACKUP_LOG_NAME}

else

echo There were no backup directories that were more than ${expire_minutes} minutes old:

echo "[${TS}] There were no backup directories that were more than ${expire_minutes} minutes
old" >> ${BACKUP_LOG_NAME}

fi

else

if [ $expire_days -gt 0 ]; then

echo These directories are more than ${expire_days} days old and they are being removed:

else

echo These directories are more than ${expire_minutes} minutes old and they are being
removed:

fi

echo "------------------------------------------------------------------------"
```

```
echo "${expire_minutes} = ${expire_minutes} minutes"

counter=0

for del in $(find $BASE_DIR -name '*-[0-9][0-9].[0-9][0-9].[AP]M' -mmin +${expire_minutes})

do

counter=$(( counter + 1 ))

echo $del

rm -R $del

done

fi

fi

# Synchronize with RSYNC Folder

if [ $ACTIVATE_RSYNC == "TRUE" ]; then

echo "Now synching to '$RSYNC_DIR'"

echo "[${TS}] Now synching to '$RSYNC_DIR'" >> ${BACKUP_LOG_NAME}

# replace params to necessary parameters

mkdir -p "$RSYNC_DIR"

rsync -r -t -p -o -g -v --progress --delete -c "$BASE_DIR" "$RSYNC_DIR"

echo "[${TS}] '$BASE_DIR' is now synced with '$RSYNC_DIR'" >> ${BACKUP_LOG_NAME}

fi

# Output process time

TIME_2=`date +%s`

elapsed_seconds=$(( ( $TIME_2 - $TIME_1 ) ))

elapsed_minutes=$(( ( $TIME_2 - $TIME_1 ) / 60 ))
```

```
# just a sanity check to make sure i am not running a dump for 4 hours

echo "[${TS}] This mysql dump ran for a total of $elapsed_seconds seconds." >>
${BACKUP_LOG_NAME}

echo "[${TS}] End backup process #$countBackups." >> ${BACKUP_LOG_NAME}

echo "--------------------------------------------------------------------------" >> ${BACKUP_LOG_NAME}

cd `echo $current_dir`

echo -n "Restored working directory to : "

pwd
```

-------- end script (copy above) --------